UUCS-87-002

PPL QUICK REFERENCE GUIDE (CMOS)

Steven R. Jacobs

Kent F. Smith

January 22, 1987

VLSI RESEARCH GROUP
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF UTAH
SALT LAKE CITY, UTAH 84112
(801) 581-8580

This work was supported in part by *Defense Research Projects Agency* under Contract Number DAAK1184K0017. All opinions, findings, conclusions or recommendations expressed in this document are those of the author(s) and do not necessarily reflect the view of DARPA.

Table of Contents

1. PPL Program Summary

Descriptions of valid ways to start PPL programs, and options available.

tiler (PPL editor)

tiler
tiler module (.ppl extension assumed)
tiler module.ext (explicit extension)

simpplex (PPL circuit extractor)

simpplex (prompts user for module name)
simpplex module (.ppl extension assumed)
simpplex module.ext (explicit extension)

options:

-t produce timing warnings for "slow" nodes (optional value specifies the minimum delay time that will produce a warning, example: -t25)

-l produce warnings for nodes with loads greater than specified value This is similar to -t option, but allows specifying minimum load value will that will produce a warning.

simppl (PPL logic simulator)

simppl (prompts user for module name)
simppl module (.ppl extension assumed)
simppl module.ext (explicit extension)

pplpr (PPL print formatter)

pplpr (uses stdin, stdout)
pplpr module
pplpr module.ext

options:

-d format for Display terminal (underline when possible, unless -u)
-h do not number rows and columns
-p Postscript device, font is scaled to fit module on one page
-n Narrow device (80 columns, default if no -p option)
-w Wide device (132 columns)
-u turn off all underlining, use characters to display breaks
-x unlimited width device (default for -p option)

ppl2cif (PPL CIF generator)

options:

-e extract named modules from library (must use -l also)
-f force generation of output even if errors are found
-O optimize output by removing unused connections between cells
-m module only, do not extract cells from library
-p Pseudo-cif output format
-s source file is a .sif pseudo-cif file, translate to real CIF
-w suppress warning messages for cells defined but not used
-C Consistency check of PPL library (must use -l also)
-l Specify PPL library name, example: -lNMOS
-9 keep user extension "9" lines in CIF, used to specify cell names

2. TILER Editor Command Summary

2.1. CONTROL commands

^A move to first column on screen.
^B move Backward (left) one or more columns.
^C Suspend TILER. TILER may be re-entered without losing anything.
^D Delete PPL cell (and breaks) at current cursor location.
^E move to End of PPL (but remain on this screen).
^F move Forward (right) one or more columns.
^G Universal abort command.
^H (backspace). Delete break information at the current cursor location.
^I (Tab). Move to PPL grid location, given as two arguments.
^J No op.
^K Kill a window and surrounding breaks, place on the kill ring.
^L Repaint the screen.
^N move down to Next row.
^P move up to Previous row.
^Q Quote. Sneak past placement restrictions.
^R Reverse search for a specified cell.
^S Search for a specified cell.
^U Universal argument.
^V move down one or more screens.
^W copy a Window onto the kill ring, without deleting it.
^X command prefix for eXtended commands.
^Y Yank a block of cells from the kill ring.
^Z command prefix for META-CONTROL commands.
^[(same as <esc>). Command prefix for META commands.

2.2. META commands

M-A move to start of PPL (but remain on this screen).
M-C incremental CIF check. Move to location of 'next' CIF error.
M-D Set default Direction.
M-F Fill window with cells. TILER prompts for the cell character.
M-L Set screen Length.
M-M Modify cell. Set the modifier value of the cell to the given argument.
M-O Move to window. Same as M-[
M-Q Toggle QUOTE mode. Always sneak past placement restrictions.
M-R Toggle RIGID mode.
M-T set Terminal display mode. Set display mode used for break characters.
M-U Toggle USER mode.
M-V move up one or more screens.
M-W Set screen Width.
M-Y Yank again, rotating through the kill ring.
M-Z Toggle ZOOM mode. (Not yet implemented)

M-< move to top edge of PPL array, remaining in the same column.
M-> move to bottom edge of PPL array, remaining in the same column.
M-[Move to WINDOW (lower left corner).
M-] Move to WINDOW (upper right corner).
M-{ Set lower-left corner of WINDOW. Argument between 0 and 9 (Default 1).
M-} Set upper-right corner of WINDOW. Argument between 0 and 9 (Default 1).
M-~ Tell TILER to pretend that the current file has not been modified.
M-;,
M-: Toggle COORDINATE ECHO mode. Turns on/off the Rxx:Cxx in status line.
M-/,
M-? Describe a TILER command or PPL cell. You type the command characters.

2.3. META-CONTROL commands

(use ^Z prefix, or <esc>-<control character>)
^Z-B move Backward (left) one or more screens. May be preceded by argument.
^Z-C Suspend TILER. TILER may be re-entered without losing anything.
^Z-D Delete PPL cell at current cursor location, but keep surrounding breaks.
^Z-F move Forward (right) one or more screens. May be preceded by argument.
^Z-G (aborted command, rings bell only)
^Z-I (META Tab). Set MARK. Argument between 0 and 9. (Default 1)
^Z-K Delete (cells from) WINDOW, but keep surrounding breaks. (Arg 0 to 9).
^Z-L Adjust screen, put cell under the cursor at upper-left corner of screen.
^Z-N move down one or more screens. May be preceded by an argument.
^Z-O move to origin of PPL array, changing screens if necessary.
^Z-P move up one or more screens. May be preceded by an argument.
^Z-W Write WINDOW (to PPL file). Argument between 0 and 9 (default 1).
^Z-Z Suspend TILER. TILER may be re-entered without losing anything.
<esc><esc>,
M-<esc>,
M-^[,
^Z-[(<esc><esc>) Escape to/from USER mode. (Prefix to switch keyboards)
^Z-_,
^Z-/
^Z-? Display the last error message encountered.

2.4. eXtended commands

^X-A move to Start of PPL. Moves to the first PPL cell or break on the row.
^X-B select Buffer. Switch to buffer with the specified name.
^X-C CIF checks. Creates .ERR file listing cell placement errors.
^X-D Define key. Assigns the current keyboard macro to the specified key.
^X-E Execute keyboard macro. May be preceded by an argument (repeat count).
^X-F list File. Displays contents of a text file on the screen.
^X-H Mark whole buffer. Selects the entire PPL array as the default WINDOW.

^X-I Insert PPL file. TILER prompts for the file name.
^X-K Describe User Key. TILER prompts for key, shows keystroke definition.
^X-L show Location in PPL buffer. Useful when coordinate echo is off.
^X-M Modify window. Sets the modifier value of cells in the current window.
^X-O Move to PPL Origin. Place cursor at row 0, column 0.
^X-T new Technology. Start a new edit session with a different cell set.
^X-U Undefine key. Removes a key definition from the specified key.
^X-X eXecute a TILER command file (usually created by logging with ^X^L).
^X-= show Location in PPL buffer. Useful when coordinate echo is off.
^X-(Begin defining keyboard macro.
^X-) End of keyboard macro.
^X-[Move to origin (lower-left corner) of the PPL array.
^X-] Move to the upper-right corner of the PPL array.
^X-{ Set lower-left corner of PPL. CAUTION: Reduces size of PPL array.
^X-} Set upper-right corner of PPL. CAUTION: Reduces size of PPL array.
^X-/,
^X-? HELP!
^X^A move Backwards to the left edge of the PPL array.
^X^B list Buffers. Lists names of all buffers and associated files.
^X^C Exit TILER. Offers to save the file if changes have been made.
^X^E move to End of PPL. Move to the extreme right edge of the PPL array.
^X^F Find file. Edit a new file in its own buffer (or return to buffer).
^X^G (aborted command, rings bell only)
^X^I Insert PPL rows or columns (Argument specifies how many).
^X^K Delete PPL rows or columns (Argument specifies how many).
^X^L Log TILER commands. Start/pause/resume/stop logging TILER commands.
^X^O move to PPL origin (row 0, column 0).
^X^R Read PPL file. Edit a new file, without changing to a new buffer.
^X^S Save changes in the current file.
^X^V Visit file. Same as ^X^F. (Visit PPL file).
^X^W Write file. TILER prompts for the file name.
^X^X Exchange POINT and MARK. Argument between 0 and 9 (Default 1).
^X^Z Suspend TILER. TILER may be re-entered without losing anything.

3. SIMPPL Simulator Command and Option Summary

3.1. Summary of SIMPPL commands

BREAK	Pause in execution of a command file.
CLOCK	Specify clock sequences associated with input nodes.
CONTINUE	Resume execution of a command file (after a BREAK).
COPY	Copy simulation output to a file.
CYCLE	Continue circuit simulation for a specified number of clock cycles.
EXIT	Exit the simulator program.
FORCE	Specify nodes to be forced to present (or specified) values.
HELP	Print helpful information about commands and options.
INIT	Initialize all circuit nodes to same state (0, 1, or X).
MAP	Find the node number of a specified PPL grid location.
NAME	Name a circuit node (specified by PPL grid location).
NODE	Name a circuit node (specified by node number).
OPTIONS	Set simulator options.
PAD	Name a pad cell (specified by PPL grid location).
PHASE	Continue circuit simulation for a specified number of clock phases.
PROBE	Name internal nodes of cell at a specified PPL grid location.
PROMPT	Change the prompt issued by the simulator when ready for commands.
QUIT	Exit one level of simulation (end of command file, or exit).
RESTORE	Restore the circuit state from a file.
SAVE	Save the present state of the circuit in a file.
SET	Set the state value of input nodes and/or forced nodes.
SHOW	Show the current state of the specified nodes or the WATCH list.
SOURCE	Take simulator commands from a specified command file.
STATUS	Show state of a node, and interaction with other nodes.
STEP	Resume simulation for a specified number of unit steps.
TRACE	Add the specified nodes to the TRACE list.
TRAP	Add the specified nodes to the TRAP list.
UNCLOCK	Remove the specified nodes from the CLOCK list.
UNFORCE	Remove FORCE status from the specified nodes.
UNTRACE	Remove the specified nodes from the TRACE list.
UNTRAP	Remove the specified nodes from the TRAP list.
UNWATCH	Remove the specified nodes from the WATCH list.
VECTOR	Group several nodes together with a single name.
WATCH	Add the specified nodes to the WATCH list.
WIRES	Show the wire names used in the current cell set.
?	Give a list of available commands.

3.2. Summary of SIMPPL options

ASYNCH	Use the asynchronous model for simulation.
ECHO	Turn on/off echo of commands executed from a command file.
SHOWINPUTS	Show the initial state of the WATCH list when resuming simulation.
SYNCH	Use the synchronous model for simulation.
TRACE	Enable/disable tracing of nodes on the TRACE list.
TRAP	Enable/disable trapping of nodes on the TRAP list.
VERBOSE	Control amount of simulation data that is printed.

4. CMOS Cell Set Quick Reference

4.1. CMOS Cell Set "Outline"

4.1.1. Basic PPL Cells

1) Wires, breaks, routing cells
   A) BLANK cell (wires everywhere)
   B) BREAK cells (breaking wires)
   C) Row/column interconnect cells
2) PAD cells
   A) Simple inputs and outputs ('K', 'Q')
   B) Tri-state I/O pad ('Z')
   C) Fill cell ('')
   D) VDD and GND pads ('P', 'G')
3) Gates and Combinatorial logic
   A) Inverters ('i', 'v')
   B) AND elements ('1', '0')
   C) OR elements ('+', 'r', 's')
4) Synchronous Flip Flops
   A) JK flip-flops ('F')
   B) D flip-flops ('D')

4.1.2. Intermediate PPL Cells

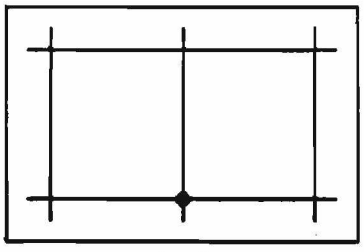
1) Universal Interconnect
   A) '?' cell
2) More inverters and buffers
   A) 'I', 'V', 'J', 'W' inverters and buffers
   B) '>', '<' Inline Row Buffers
3) AND-OR and OR-AND elements
   A) 'E', 'X', 'e', 'x'
4) Multiplexor cells
   A) 'h', 'H'
6) Transparent Latch
   A) 'L'
6) RAM cells
   A) Data Elements and Caps ('M', '[', ']', 'n', 'o', 'p')
   B) Sense Amplifiers and Caps ('A', '{', '}')

4.1.3. Advanced PPL Cells (rarely used)

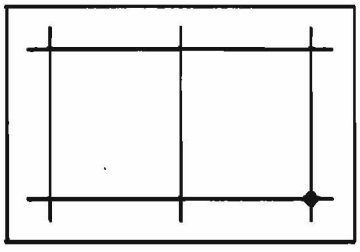
1) 'g', '^'
2) '(', ')', '\', '/'

4.2. CMOS Cell Schematics

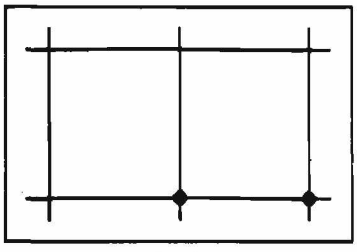
(See pages 11 through 22
for cell schematics)



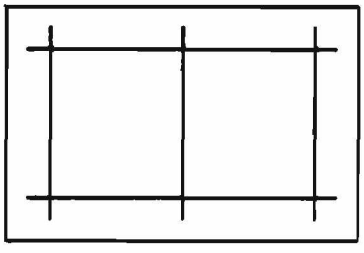
'*'



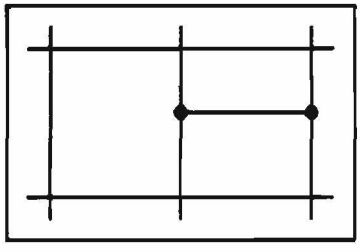
'#'



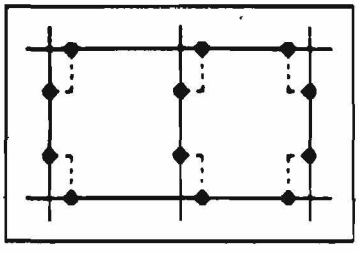
'@'



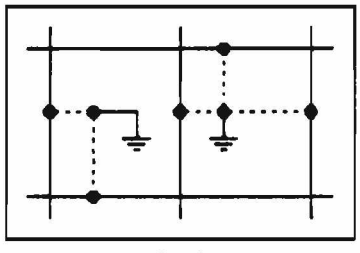
' '



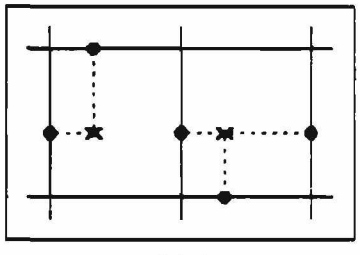
'~'



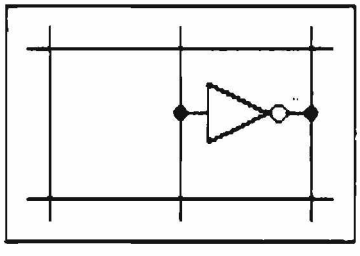
'?'



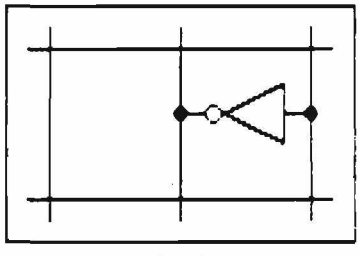
'g'



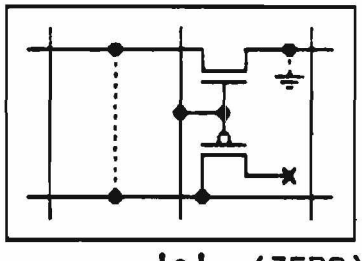
'^'



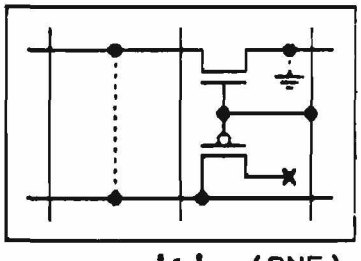
'v'



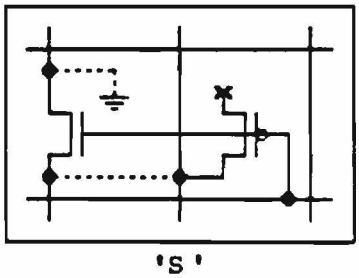
'i'



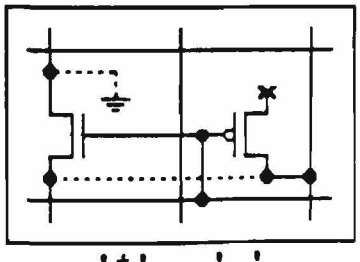
'0' (ZERO)



'1' (ONE)

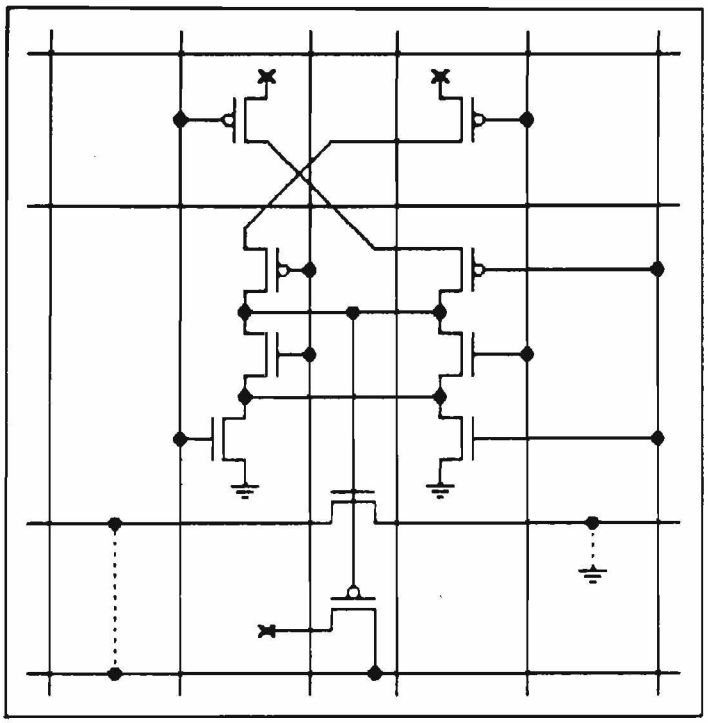


's'

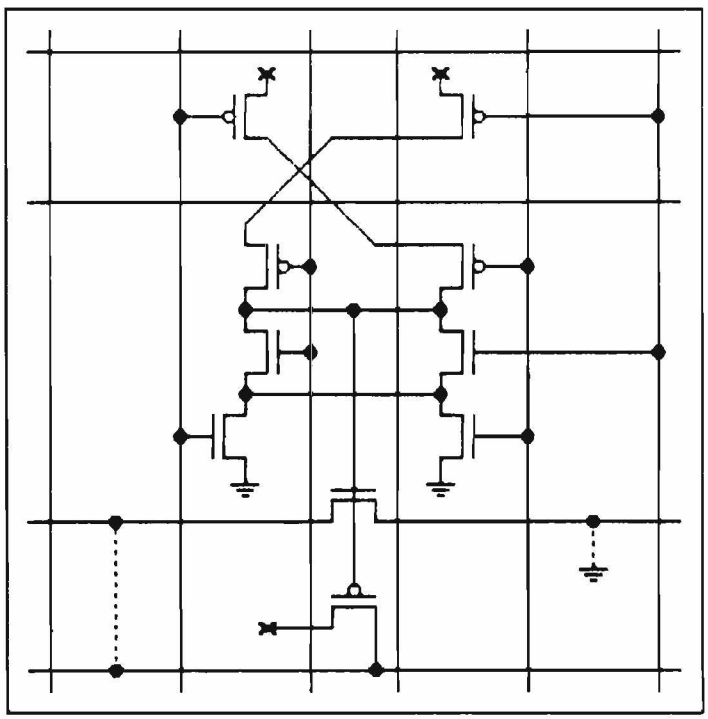


'+' , 'r'

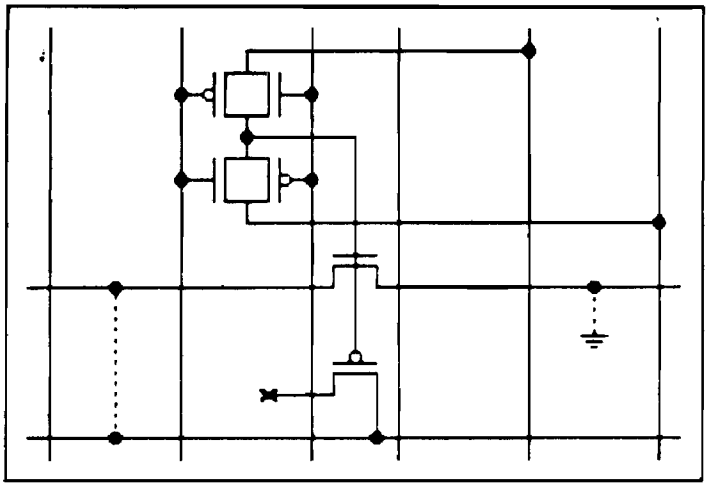
'I'
'V'
'<'
'J'
'W'
'>'



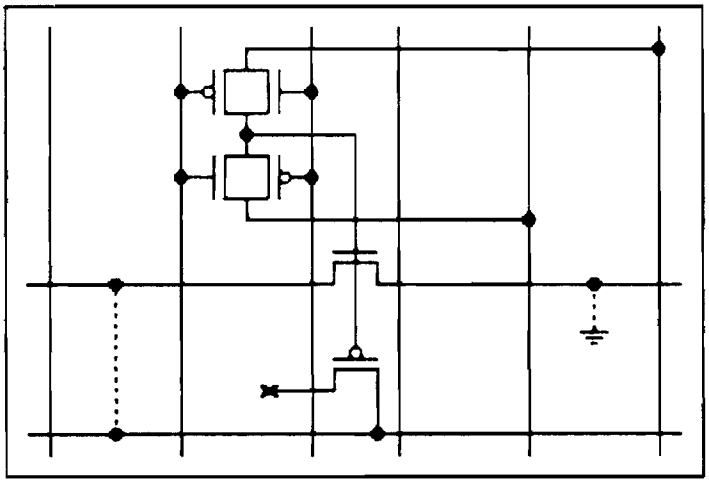
'X'



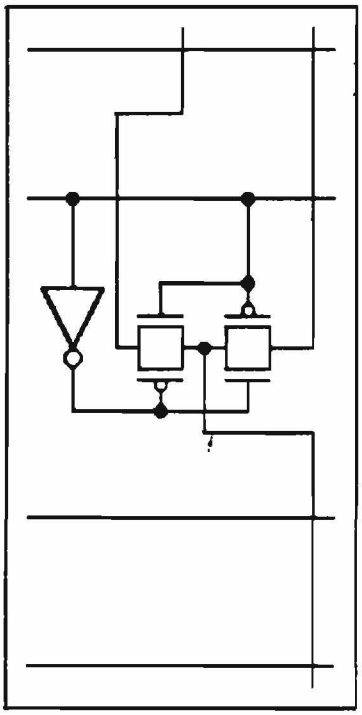
'E'



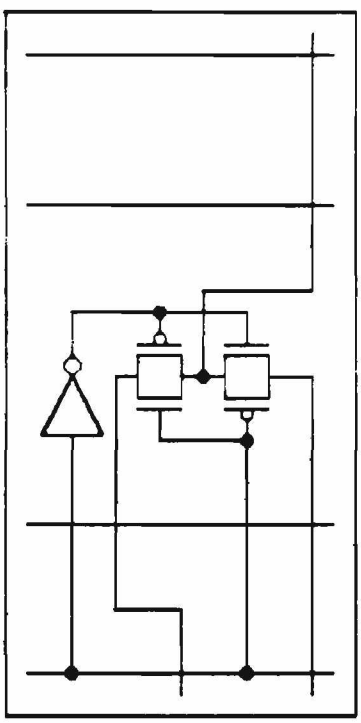
'x'



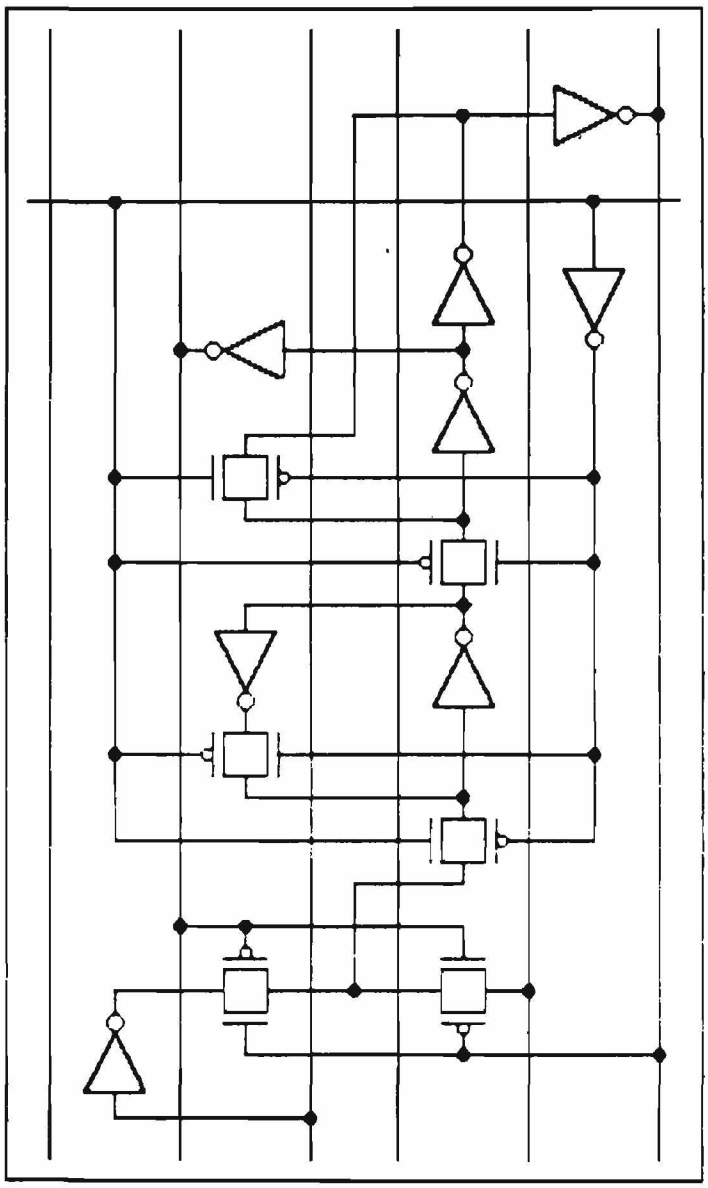
'e'



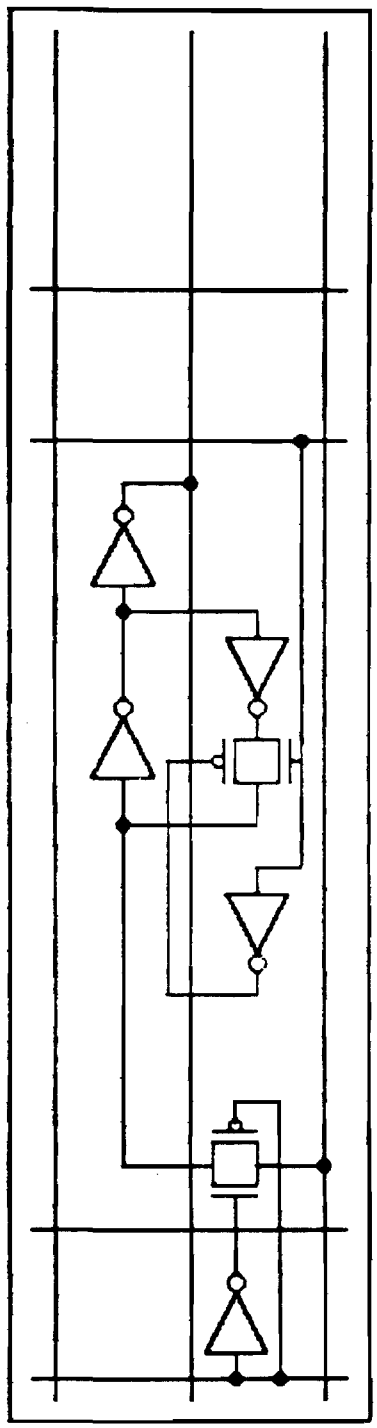
'H'



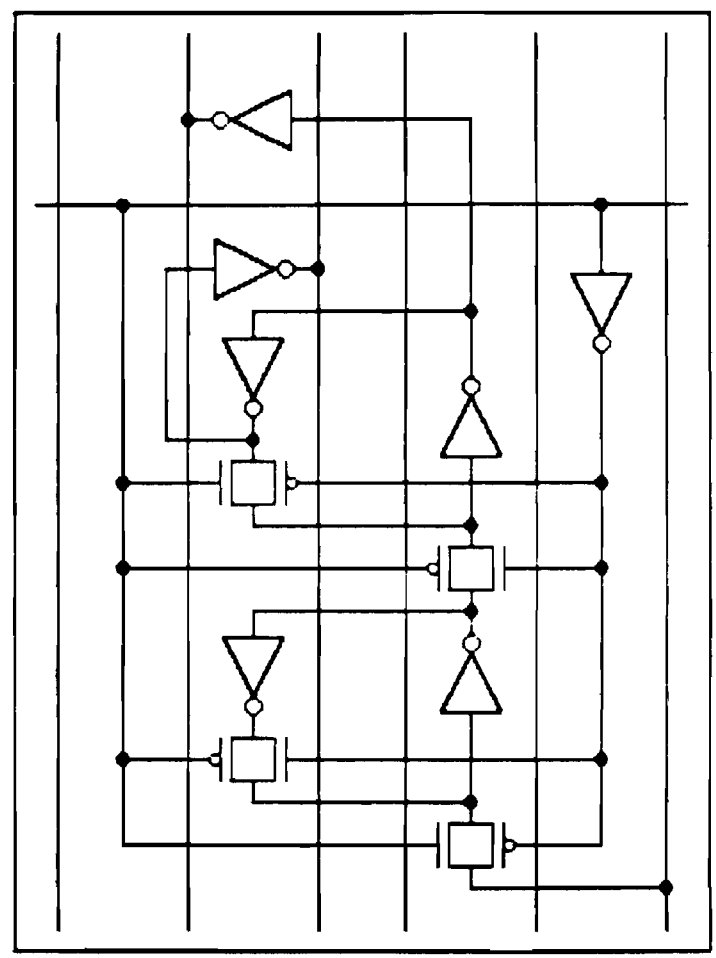
'h'



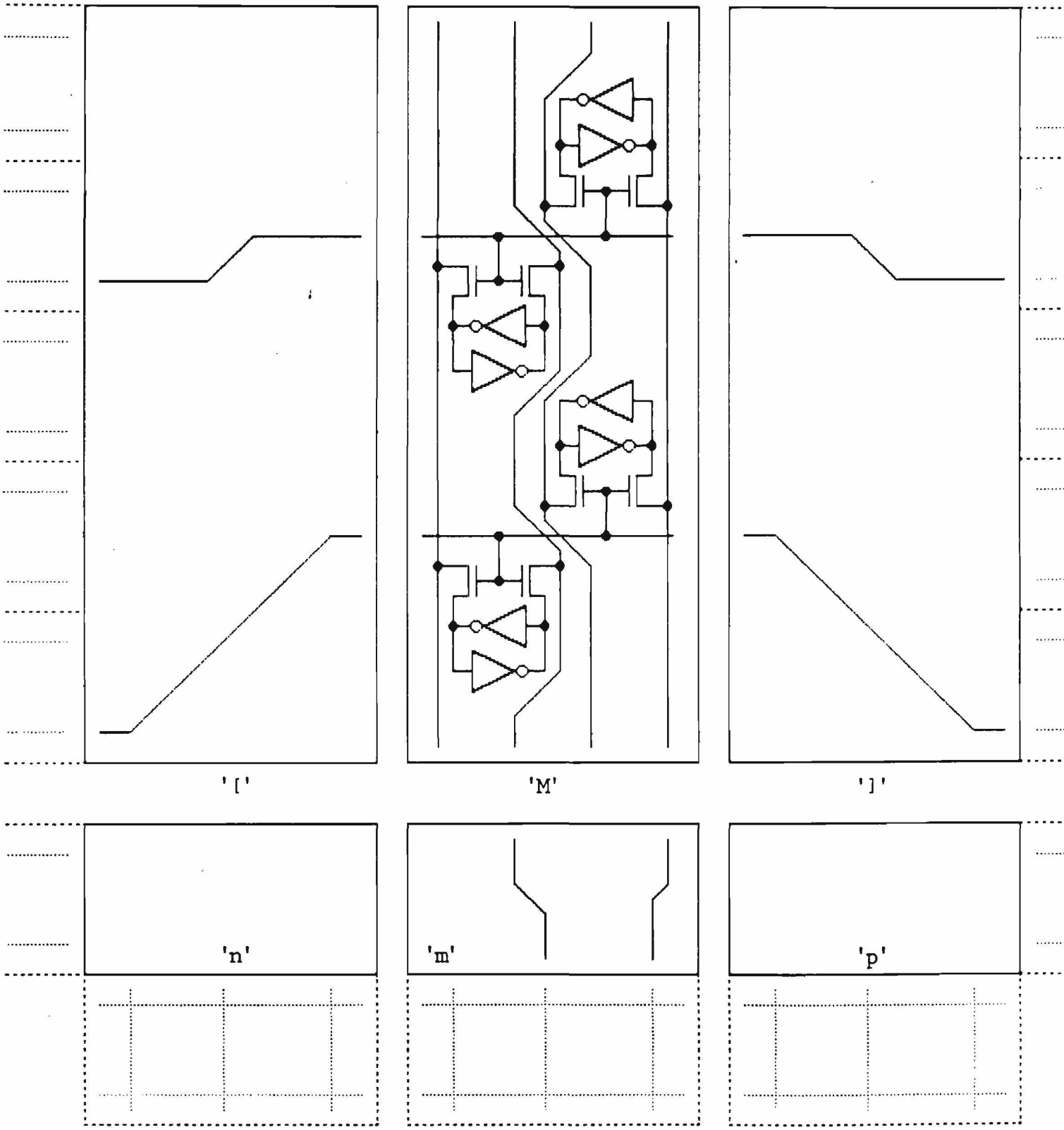
'F'

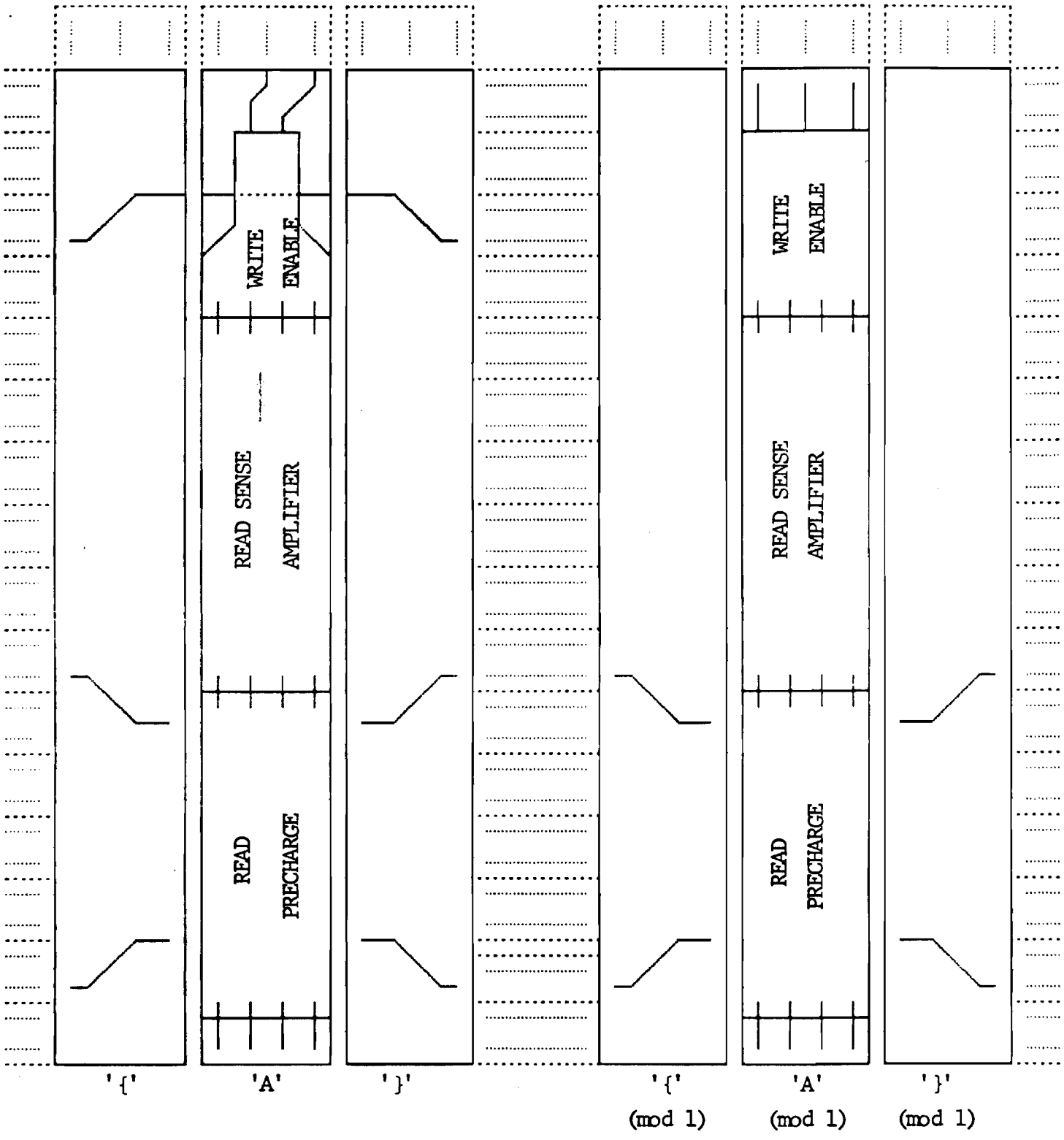


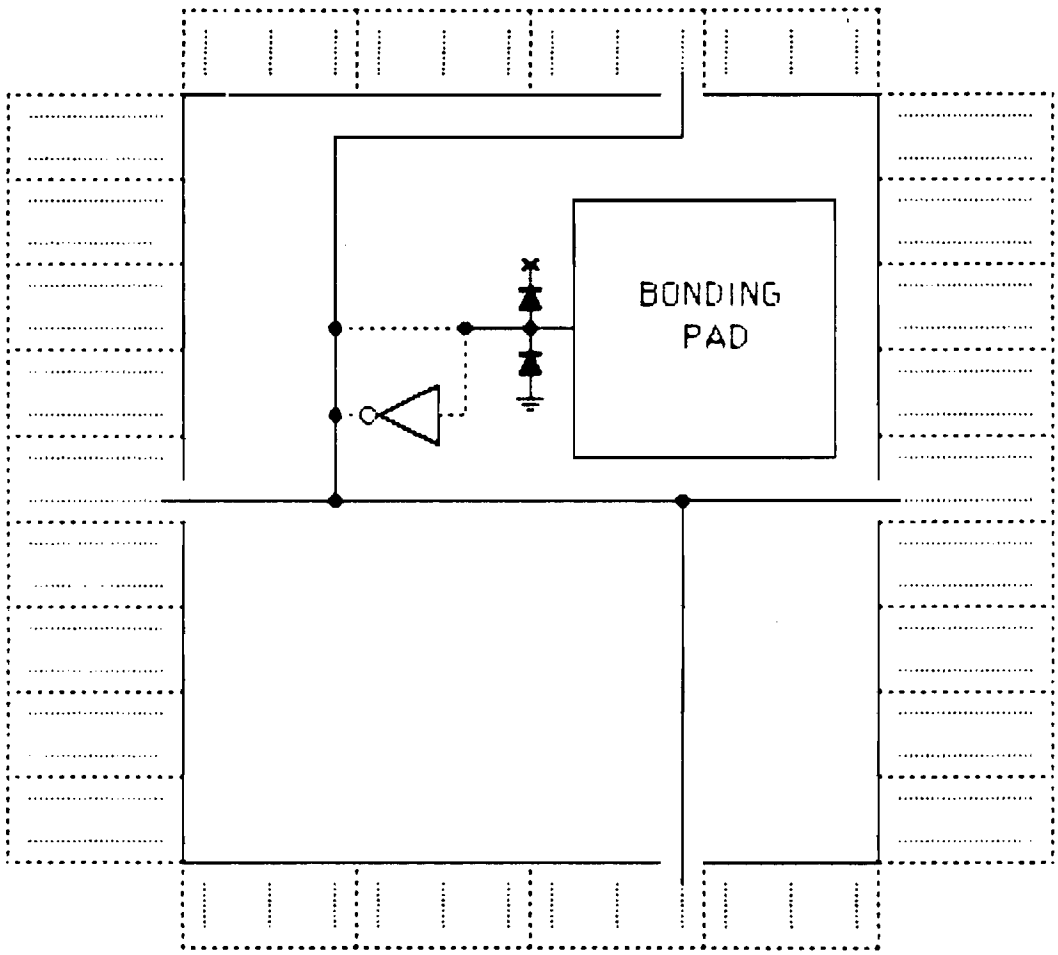
'L'



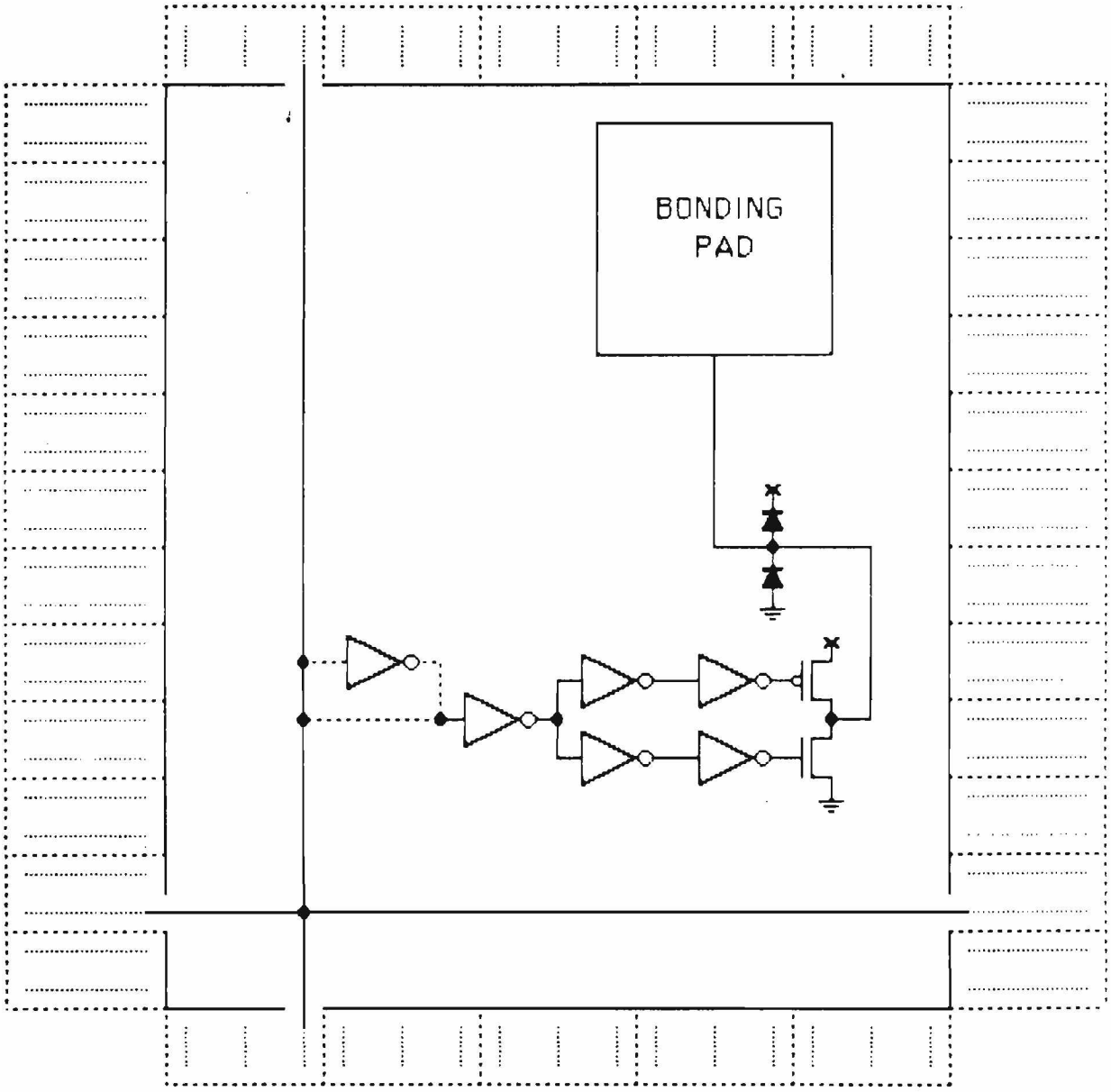
'D'


'['
'M'
']'
'n'
'm'
'p'

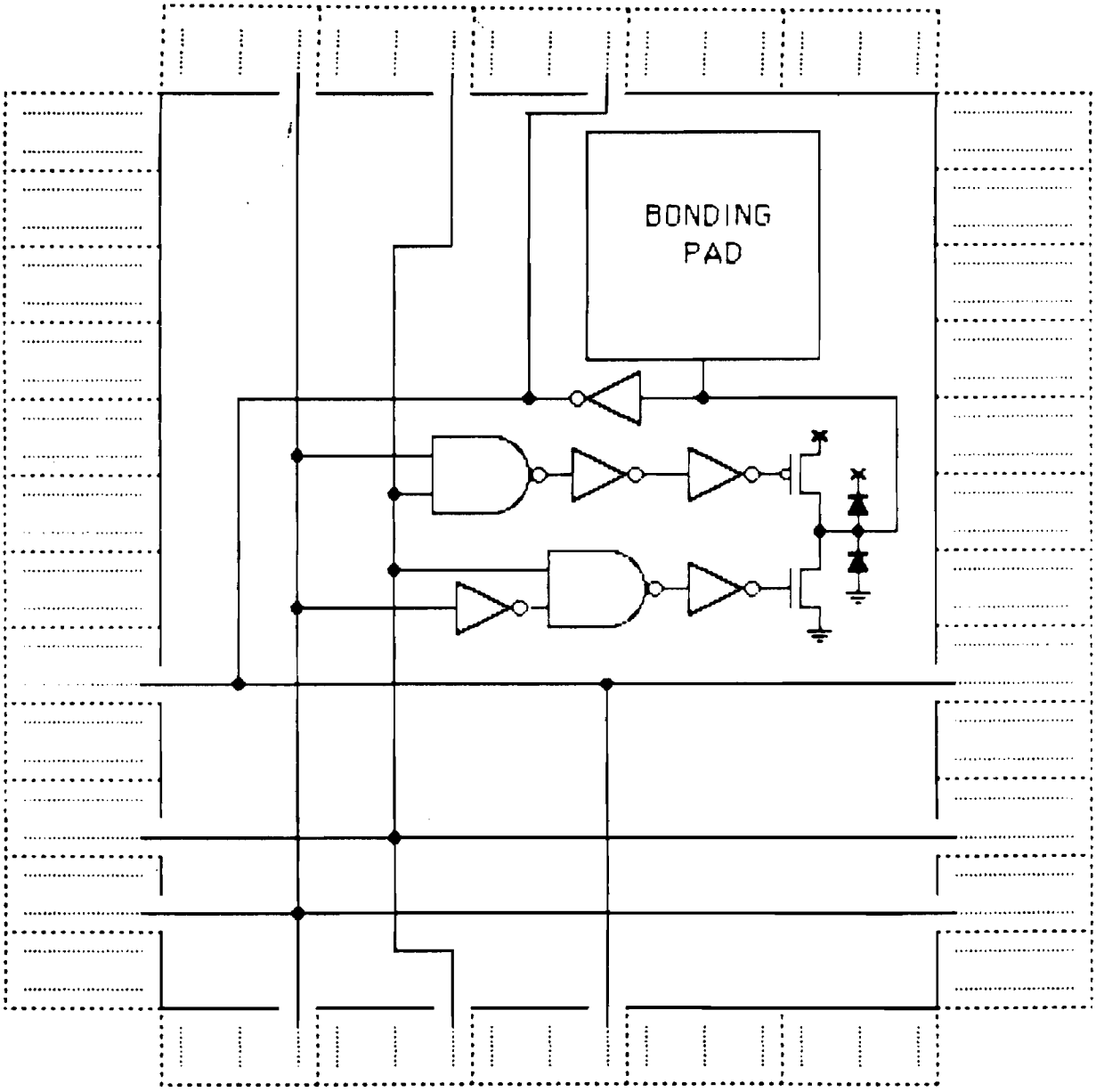

WRITE ENABLE
READ SENSE AMPLIFIER
READ PRECHARGE
'{'
'A'
'}'
'{' (mod 1)
'A' (mod 1)
'}' (mod 1)



'K'



'Q'



'Z'

5. The PPL Methodology

Path Programmable Logic (PPL) is a cell based design system. When designing PPL circuits, a special text editor is used to place cells (represented by characters) on a regular rectangular grid. The PPL grid consists of many regularly spaced wires running vertically (column wires) and horizontally (row wires). A wire in the PPL grid normally extends the full height (or width) of the PPL array without connecting to any of the wires which run perpendicular to it. Special "routing" cells may be used to connect row wires to column wires. When cells that contain active devices are placed in the PPL array, the inputs and outputs of the cell are automatically connected to grid wires. Cells are connected together simply by positioning them so that they connect to the same grid wires. Cells placed in the same row are connected by the row wires of the grid, and cells placed in the same column are connected by the column wires of the grid. BREAKS may be placed between cells to split grid wires, so that connections between neighboring cells may be broken. Breaks are displayed in the editor as '|' and '_' characters. The '|' character indicates a broken row wire, and an underline indicates a broken column wire. Breaks that have been inserted into the PPL array may be removed by using the appropriate commands in the PPL editor.

PPL cells are almost always rectangular in shape, and always occupy a whole number of grid locations. The size of PPL cells is usually expressed in terms of rows and columns. Many of the cells require only a single grid location, meaning that they occupy one row and one column. Other cells may occupy a rectangle that is several rows high and several columns wide, depending on the complexity of the cell. The physical size of a single PPL row or column is fixed and does not vary between cells. Each PPL column will usually contain at least two column wires, and these are referred to as the "left" column (LCOL) and "right" column (RCOL) wires. Pairs of wires are used so that a signal and its compliment may be generated in the same column. PPL rows may contain one or more row wires, depending on the cell set. Complimentary pairs of signals are not usually created on row wires. Some cell sets may have a "primary" row wire which is used for signal inputs and outputs, along with a "secondary" row wire which may only be used for routing signals between columns.

When cells are placed in the PPL array, the pieces of circuitry in the cells can be thought of as being placed "under" the wires that make up the PPL grid. PPL cells usually cause

some kind of interaction between signals on grid wires that pass through the cell. For example, a PPL cell might cause one of its row wires to be forced to a low voltage whenever a high voltage appears on a certain column wire, and let all other grid wires pass through the cell unaltered. Signals on "unused" wires are usually allowed to pass through the cell unaltered. Some PPL cells do not have enough "spare room" to allow all of the grid wires to pass through the cell. These cells cause "forced" breaks in the grid wires. Forced breaks are displayed in the same manner as breaks that are inserted by the user, but forced breaks cannot be removed from the grid except by removing the cell. Cells with forced breaks can be thought of as removing pieces of wire from the PPL grid in order to make room for the circuitry in the cell.

The characters used to represent PPL cells are usually chosen to reflect the logical function of the cell. The '1' and '0' characters might be used for cells that determine whether a signal is TRUE or FALSE, and the '+' character might be used to generate a logical OR of several signals. The 'F' character might be used to represent a flip-flop, and the 's' and 'r' characters might be used for cells which cause a flip-flop to be set or reset. The particular characters used for each cell will vary from cell set to cell set. A cell set designed for an NMOS process will probably be quite different than a cell set developed for CMOS.

Choosing appropriate characters to represent PPL cells allows PPL circuits to "look like" truth tables or state transition tables. Logical AND circuits are formed by placing several '0' and '1' cells together in a single row. This row wire acts as the output of a gate that uses column wires for inputs. The row wire then becomes TRUE when all of the column wire inputs to the '0' and '1' cells are TRUE. Logical OR circuits are formed by placing '+' cells together in a single column. This column acts as the output of a gate that uses row wires for inputs. The column then becomes TRUE when one or more of the row wire inputs to the '+' cells are TRUE. In this way, AND functions are generated on rows, while OR functions are formed on columns. Examples of a PPL exclusive-or module and a PPL full adder are shown below. In these modules, inputs enter on column wires and outputs exit on column wires, and the inputs and outputs all pass through to the top and bottom of the module. The 'i' cells are inverters with inputs on the RCOL wires and outputs on the LCOL wires. Inputs to PPL modules are usually taken from the RCOL wires, because the '+' cell asserts the RCOL wire. The compilmentary signals on the column wires are

detected by the '0' and '1' cells. Row breaks are used to isolate the modules from each other. These modules look very much like truth tables, when in fact the characters also represent the actual layout of the transistors needed to implement the logic functions. When PPL circuits are designed, both the logical function and the physical implementation of the circuits are created simultaneously.

```
 i i        i i i
|0 1 +|    |0 0 1 +   |
|1 0 +|    |0 1 0 +   |
           |1 0 0 +   |
           |1 1 1 +   |
           |   1 1    +|
           |1    1    +|
           |1 1       +|
```

6. The CMOS PPL Cell Set

6.1. Mixed Logic in PPL

The logical "meaning" that is associated with the physical voltages on a node determines the type of logic system in use. Nodes which use a HIGH voltage to represent a TRUE logic value (high-true) and a LOW voltage for a FALSE logic value are said to use POSITIVE logic. Nodes which use a LOW voltage as a TRUE logic value (low-true) and a HIGH voltage as a FALSE logic value are said to use NEGATIVE logic. When the terms NAND and NOR are used in referring to gates, positive logic is implied so that NAND and NOR refer to the electrical function of the gate. Thus a "NAND gate" is a circuit that has a low voltage on its output whenever all of its inputs are at a high voltage, and a "NOR gate" is a circuit that has a low voltage on its output whenever any of its inputs is at a high voltage.

PPL cell sets are based on a "mixed" logic system. This is done so that the AND-OR structures described above may be physically implemented using only NAND gates or only NOR gates, but not both. The particular type of logic gate used depends on which process is used to fabricate the circuit (NMOS, CMOS, GaAs, etc.), and is based on circuit performance considerations. In general, NMOS circuits which are based on NOR gates can operate at higher speeds than similar NMOS circuits which are based on NAND gates. Thus NOR gates are preferred for circuits fabricated in NMOS. In most CMOS processes, NAND gates are preferred so that the faster n-channel transistors may be placed in series while the slower p-channel transistors are connected in parallel. This configuration can usually operate at higher speeds than the corresponding NOR gate which is constructed with series p-channel transistors and parallel n-channel transistors.

Cell sets which are based on electrical NOR gates (output low when any input is high) usually use positive logic on row wires while using negative logic on column wires. Cell sets which are based on electrical NAND gates (output low when all inputs are high) usually use positive logic on column wires while using negative logic on row wires. This allows a logical AND operation to be implemented by placing several '0' and '1' cells on the same row, and allows a logical OR operation to be implemented by placing several '+' cells on the same column. "Truth table" design of logic functions is performed in the conventional sum-of-products format found in most texts on logic design.

When describing the functions performed by the various cells in a PPL cell set, it is most convenient to describe the LOGICAL FUNCTION of the cells in a manner that is independent of the physical voltages used to represent the logic values involved. The cell descriptions which follow are usually worded in terms of the logical values (TRUE, FALSE) of the inputs and outputs of the cells rather than in terms of the electrical values (high, low) of the inputs and outputs. Since PPL designs use a mixed logic system, knowing the voltage of a node is not equivalent to knowing the logic state of the node. On some nodes, a high voltage is used to represent TRUE, while on other nodes a high voltage is used to represent FALSE. Since there is no simple way to derive the logical state of a node from its voltage level, it would be confusing to try to describe the logical operation of the cells in terms of voltages. Occasionally, functional descriptions of cells are given in terms of voltages in order to clarify the operation of the cell under discussion. Functional descriptions of PAD cells will also include electrical information so that a proper interface to the "outside world" can be created.

The CMOS30 cell set is based on CMOS NAND gates. Most logic gates are formed by parallel connected p-type pullup transistors which are connected to a stack of series connected n-type transistors which form a pulldown. This basic NAND gate is formed whenever a group of '0' and '1' cell are placed on a common row segment, or whenever a group of '+', 'r', or 's' cells are placed on a common column segment.

In the CMOS30 cell set, negative (low-true) logic is mostly used on row wires, while positive (high-true) logic is mostly used on column wires. This implies that cells which affect the logic state of a row wire can either force the row wire to a FALSE logic value (high voltage), or allow the row wire to remain at a TRUE logic value (low voltage) if no other cells are forcing it false. In contrast, cells which affect the logic state of a column wire can either force the column wire TRUE (high voltage), or allow the column wire to remain FALSE (low voltage). The descriptions of the logical operation of the cells are phrased in terms of "forcing" signals TRUE/FALSE or "allowing" signals to remain TRUE/FALSE to help indicate the influence of other cells that are connected to the node. Also, a signal wire is said to be ASSERTED if it is in its TRUE state, regardless of the voltage levels involved. A signal is DEASSERTED or DISABLED when it is in its FALSE state.

6.2. Special Rules: Series Stack SHORTS and GROUNDS

When PPL circuits are built with the CMOS cell set, NAND gates are built by combining together several cells. For example, when several '0' and '1' cells are combined on a common row segment, a NAND gate is created. Similarly, several '+' cells can be placed in a common column segment to create a NAND gate. CMOS NAND gates consist of a group of parallel connected p-type transistors which act to pull the output to a high voltage, together with a series stack of n-type transistors which pull the output to a low voltage. In order for the gate to be complete, one end of the series stack must be connected to ground, while the other end of the series stack is connected to the parallel output wire that is shared by the p-type pullup transistors. The connection to ground is referred to as a "ground", and the connection to the parallel output is referred to as a "short". These shorts and grounds must be placed by the circuit designer in order to complete each of the NAND gates in the circuit. This can be contrasted to the NMOS cell sets, which require pullup cells to be placed in order to complete each NOR gate in the NMOS circuit.

Unlike NMOS, which often requires an explicit cell location for each pullup, the shorts and grounds needed in CMOS circuits are inserted into the circuit by using modifiers on some of the cells which make up each CMOS gate. PPL gates in the current CMOS cell set are confined to single segments of rows and columns. Modifiers are placed on the cells at the outermost "ends" of each CMOS gate. A modifier value of one represents a short of the series wire to the parallel wire, and a modifier value of two represents a ground connection to the series wires. A modifer value of three is used when the gate has only a single input (i.e. it is an inverter made from a '0', '1', 's', 'r', '+', or similar cell). Shorts are placed on the lower end of gates built in columns and on the leftmost end of gates built in rows. Grounds are placed on the upper end of gates built in columns and on the rightmost end of gates built in rows. These rules for shorts and grounds are illustrated by the following PPL circuit. The table to the right of the circuit lists the locations and values of the modifiers in the circuit. Note that the 'i' cells are complete inverters which do not require shorts or grounds.

```
   0 1 2 3 4 5 6 7 8
  -----------------
6:
5:     1 1 1
4:     1 0 1 +
3:       1 0   +
2:     0     +
1:
0:
```

Cell	Row	Col	Modifer
0	2	2	3
+	2	5	1
1	3	3	1
0	3	4	2
+	3	6	3
1	4	2	1
1	4	4	2
+	4	5	2

In the current PPL CMOS cell sets, the cells which may be combined on rows to form NAND gates are the '0', '1', 'x', 'e', 'X' and 'E' cells. The '+' and 'r' cells can be combined together in columns to form NAND gates with outputs on RCOL wires, and 's' cells can be combined together in columns to form NAND gates with outputs on LCOL wires. The 's' cell should not be combined with the 'r' or '+' cells, since the output of the 's' is not on the same wire as the outputs of the 'r' and '+' cells.

7. CMOS Cell Descriptions

7.1. Basic PPL Cells

7.1.1. Default Cells

BLANK

Character: ' '
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

The BLANK cell passes row and column wires which do not interact. This is the "default" cell that forms the wires in the PPL array. It consists of two row wires named ROW and SROW (Series ROW), and three column wires named RCOL (Right COLumn), LCOL (Left COLumn), and SCOL (Series COLumn).

7.1.2. BREAK "cells"

BREAK cells are not actually cells, because breaks do not occupy a cell location in the PPL array. Breaks are placed between cells in the PPL array, and may be placed between any two adjacent cells. Breaks are represented by characters, and when a break character is placed at a cell location, the breaks take effect on the left and lower edges of the cell. Breaks are not always displayed as the character used to insert them. The PPL editor tries to display column breaks by underlining, and row breaks by the '|' character, whenever possible. On terminals that are not capable of underlining, other characters may be used to display break information.

COLBRK

Character: '_'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) break RCOL wire (default, no modifier value)
(1) break RCOL wire.
(2) break LCOL wire.
(4) break SCOL wire.

Breaks column wires at lower edge of cell, as specified by the modifier value. Modifier values may be added to break more than one wire.

ROWBRK

Character: '|'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) break ROW wire (default, no modifier value).
(1) break ROW wire.
(2) break SROW wire.

Breaks row wires at left edge of cell, as specified by the modifier value. Modifier values may be added together to break more than one wire.

COLBRK

Character: '_'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) break RCOL wire (default, no modifier value).
(1) break RCOL wire.
(2) break LCOL wire.
(4) break SCOL wire.

Breaks the COLUMN wires at bottom edge of cell, as specified by the modifier value. Modifers may be added together to break more than one wire.

AUXROWBRK

Character: ':'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the SROW wire at left edge of cell.

LCOLBRK

Character: ',' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks left column (LCOL) wire at lower edge of cell.

RCOLBRK

Character: '.' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks right column (RCOL) wire at lower edge of cell.

BCOLBRK

Character: '=' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks RCOL and LCOL wires at lower edge of cell.

LCOLRBRK

Character: '!' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the LCOL wire and both row wires.

RCOLRBRK

Character: '%' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the RCOL wire and both row wires.

ALLBREAK

Character: '&' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks all wires at lower and left edges of cell.

7.1.3. Pseudo Cells

COVERED

Character: '"'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

The " character does not represent an actual cell. For cells which cover more than a single grid location, the " character is displayed at all grid locations covered by the cell except for the origin of the cell, which contains the character which represents the cell.

EMPTY

Character: '–'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

The empty cell prevents insertion of any layout data at the occupied grid location. This must not be confused with the BLANK cell, which contains row and column wires as well as power supply connections. The empty cell may be used to reserve space for insertion of custom designs into the PPL array.

7.1.4. Routing Cells

Routing cells are used to make connections between row and column wires. The connections allowed by cells in this section are adequate to handle most simple routing of signals between modules. Advanced users may want to use the '?' cell (universal intersection), which is a more flexible cell that allows complex connections to be 'built' with appropriate modifier values.

XTL

Character: '*'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Intersection (ohmic contact) between the ROW wire and the LCOL wire. This cell may be used for routing signals in the PPL array.

XTR

Character: '#'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Intersection (ohmic contact) between the ROW wire and the RCOL wire. This cell may be used for routing signals in the PPL array.

XTLTR

Character: '@'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Intersection between the ROW wire and the LCOL and RCOL column wires. This cell may be used for routing signals in the PPL array.

COLSHORT

Character: '~'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Short together the Left and Right column wires. This cell may be used for routing signals in the PPL array.

7.1.5. Pad Cells

IBUFPAD

Character: 'K'
Cell size: 9 rows, 4 columns. (CMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: (0) normal (non-inverting) input pad.
(1) inverting input pad.

Inputs: P (external pad connection)
Outputs: Q (RCOL wire of second column to right of cell origin, or ROW wire of fourth row above cell origin)

Input buffer pad.

OBUFPAD

Character: 'O'
Cell size: 12 rows, 5 columns. (CMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: (0) normal (non-inverting) output pad.
(1) inverting output pad.

Inputs: D (RCOL wire of cell origin,
or ROW wire of first row above cell origin)
Outputs: P (external pad connection)

Output buffer pad.

TSIOPAD

Character: 'Z'
Cell size: 12 rows, 5 columns. (CMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: ENABLE (RCOL wire of first column to right of cell origin,
or ROW wire of second row above cell origin)
Dout (RCOL wire of cell origin,
or ROW wire of first row above cell origin)
P (external pad connection)
Outputs: Din (RCOL wire of second column to right of cell origin,
or ROW wire of fourth row above cell origin)
P (external pad connection)

This pad combines a non-inverting tri-state output buffer pad with an inverting input buffer pad. The inverting input buffer is active at all times, and data at the Din connection is inverted with respect to data on the pad itself. When the ENABLE input is held high, the non-inverting output buffer is enabled so that data on the Dout input is placed on the external pad. When the ENABLE input is low, the output buffer is disabled so that external devices may place data on the pad.

External drive capability of the output buffer is one TTL load.

FILL

Character: "" (single quote)
Cell size: 1 row, 1 column.
Placement: Edges of circuit only.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Fill in between pad cells on edges of circuit. Must be placed in all locations that are not occupied by a pad cell.

GNDPAD

Character: 'G'
Cell size: 9 rows, 4 columns.
Placement: Left/Bottom Edges of circuit only, corner allowed.
Modifiers: None.

Inputs: (none)
Outputs: (none)

GND power supply bonding pad.

VDDPAD

Character: 'P' (single quote)
Cell size: 9 rows, 4 columns.
Placement: Upper/Right Edges of circuit only, corner allowed.
Modifiers: None.

Inputs: (none)
Outputs: (none)

VDD power supply bonding pad.

7.1.6. Basic Logic Elements

INV1

Character: 'i'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: A (RCOL wire)
Outputs: Q (LCOL wire)

The Q output remains true until the A input becomes true.

INV0

Character: 'v'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: A (LCOL wire)
Outputs: Q (RCOL wire)

The Q output remains true until the A input becomes true.

ONE

Character: '1'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: ONE (RCOL wire)
Outputs: Q (ROW wire)

If ONE is true, Q is allowed to remain true. Otherwise, Q is made false. Other cells connected to the Q output may influence the value of the Q output. In particular, other '0' and '1' cells placed on the same row produce a logical AND of the column wires which act as inputs to these cells. Other cells such as the 'E' and 'X' cells may be combinded with the '0' and '1' cells to produce more complex OR-AND type logic elements.

ZERO

Character: '0' (zero)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: ZERO (LCOL wire)
Outputs: Q (ROW wire)

If the ZERO is true, Q is allowed to remain true. Otherwise, Q is made false. Other cells connected to the Q output may influence the value of the Q output. In particular, other '0' and '1' cells placed on the same row produce a logical AND of the column wires which act as inputs to these cells. Other cells such as the 'E' and 'X' cells may be combined combined with the '0' and '1' cells to produce more complex OR-AND type logic elements.

PLUS

Character: '+'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add SCOL/RCOL short on bottom of cell.
(2) add SCOL wire ground on top of cell.
(3) add both short and ground on SCOL wire.

Inputs: A (ROW wire)
Outputs: Q (RCOL wire)

Same as 'r' cell. If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several '+' cells placed in the same column produce a logical OR of the rows which act as inputs to the '+' cells.

RESET

Character: 'r'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add SCOL/RCOL wire short on bottom of cell.
(2) add SCOL wire ground on top of cell.
(3) add both short and ground on SCOL wire.

Inputs: A (ROW wire)
Outputs: Q (RCOL wire)

Same as '+' cell. If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several 'r' cells placed in the same column produce a logical OR of the rows which act as inputs to the 'r' cells.

SET

Character: 's'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add SCOL/LCOL wire short on bottom of cell.
(2) add SCOL wire ground on top of cell.
(3) add both short and ground on SCOL wire.

Inputs: A (ROW wire)
Outputs: Q (LCOL wire)

If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several 's' cells placed in the same column produce a logical OR of the rows which act as inputs to the 's' cells.

7.1.7. Synchronous Flip-flops

JK

Character:	'F'
Cell size:	5 rows, 2 columns.
Placement:	Any row, ODD column.
Modifiers:	None.
Inputs:	SET (LCOL wire of right column)
	RESET (RCOL wire of left column)
	CLK (ROW wire of upper row)
Outputs:	Q (RCOL of right column)
	Qnot (LCOL wire of left column)

This synchronous edge-triggered JK flip flop changes its outputs when the CLK input becomes true. The flip flop will become set (Q true and Qnot false) if the SET input is asserted during CLK, and will become reset (Q false, Qnot true) if the RESET input is asserted during CLK. If both SET and RESET are asserted during CLK, the flip flop will toggle state.

DFF

Character:	'D'
Cell size:	4 rows, 2 columns.
Placement:	Any row, ODD column.
Modifiers:	None.
Inputs:	D (RCOL wire of right column)
	CLK (ROW wire of top row)
Outputs:	Q (RCOL wire of left column)
	Qnot (LCOL wire of left column)

This synchronous edge-triggered D flip flop changes its outputs when the CLK input becomes true. The flip flop stores the data from the D input on the true-going edge of the CLK input.

7.2. Intermediate PPL Cells

UNICON

Character: '?'
Cell size: 1 row, 1 column.
Placement: ANY row, ANY column.
Modifiers: [formed by summing values below]
(1) connect RCOL wire with ROW wire.
(2) connect LCOL wire with ROW wire.
(4) connect SCOL wire with ROW wire.
(8) connect RCOL wire with SROW wire.
(16) connect LCOL wire with SROW wire.
(32) connect SCOL wire with SROW wire.

Inputs: (none)
Outputs: (none)

Universal interconnect cell. Modifier value determines connections between the row and column wires, and/or internal breaks of row and column wires. Used for complex routing connections where ordinary '*', '#', '~', and '@' cells are not adequate. A modifier value of ZERO causes no internal connections and no internal breaks, which is equivalent to a BLANK cell. Various combinations of internal connections and internal breaks may be formed by summing the modifier values listed above.

NOTE: Because of cell mirroring, internal wire breaks (modifier values greater than 32) should not be used when the cell is placed in EVEN rows or EVEN columns. (Internal wire breaks are not yet implemented).

7.2.1. Inverters and Buffers

INVERT1

Character: 'I'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (none)

Inputs: A (RCOL wire)
Outputs: Q (LCOL wire)

Medium sized interting buffer. The Q output remains true until the A input becomes true.

INVERT0

Character: 'V'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (none)

Inputs: A (LCOL wire)
Outputs: Q (RCOL wire)

Medium sized interting buffer. The Q output remains true until the A input becomes true.

BIGBUF1

Character: 'J'
Cell size: 4 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: A (RCOL wire)
Outputs: Q (LCOL wire)

Large sized interting buffer. The Q output remains true until the A input becomes true.

BIGBUF0

Character: 'W'
Cell size: 4 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: A (LCOL wire)
Outputs: Q (RCOL wire)

Large sized interting buffer. The Q output remains true until the A input becomes true.

ROWINVRL

Character: '<'
Cell size: 2 rows, 1 column.
Placement: Any row, odd column.
Modifiers: (none)

Inputs: A (ROW wire on right edge)
Outputs: Q (ROW wire on left edge)

Medium sized in-line row wire inverting buffer. This cell is usually used to buffer the PRECHARGE and ENABLE inputs of the RAM sense amp cells.

ROWINVLR

Character: '>'
Cell size: 2 rows, 1 column.
Placement: Any row, odd column.
Modifiers: (none)

Inputs: A (ROW wire on left edge)
Outputs: Q (ROW wire on right edge)

Medium sized in-line row wire inverting buffer. This cell is usually used to buffer the PRECHARGE and WRITE-ENABLE inputs of the RAM sense amp cells.

7.2.2. Intermediate Logic Elements

EQUIVA

Character: 'E'
Cell size: 3 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: A (RCOL wire of left column)
A* (LCOL wire of left column)
B (RCOL wire of right column)
B* (LCOL wire of right column)
Outputs: Q (lower ROW wire)

The Q output is left true if (A and B) is true, or (A* and B*) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an equivalence function on the A and B columns. However, the cell can perform more general and-or type logic when four distinct signals are used for inputs. In particular, several 'E' cells placed side by side form an arithmetic comparator which leaves Q true whenever the two input values are equal. Other cells connected to the Q output may influence the value of the Q output.

EXORA

Character: 'X'
Cell size: 3 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: A (RCOL wire of left column)
A* (LCOL wire of left column)
B (RCOL wire of right column)
B* (LCOL wire of right column)
Outputs: Q (lower ROW wire)

The Q output is left true if (A and B*) is true, or (A* and B) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an exclusive-or function on the A and B columns. However, the cell can perform more general and-or type logic when four distinct signals are used for inputs. Other cells connected to the Q output may influence the value of the Q output.

EQUIVB

Character: 'e'
Cell size: 2 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: A (RCOL wire of left column)
A* (LCOL wire of left column)
B (RCOL wire of right column)
B* (LCOL wire of right column)
Outputs: Q (lower ROW wire)

The Q output is left true if (A and B) is true, or (A* and B*) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an equivalence function on the A and B columns. However, the cell can perform more general and-or type logic when two distinct signals are used for the B and B* inputs. Other cells connected to the Q output may influence the value of the Q output.

NOTE: The A and A* signals MUST be logical inverses in order for this cell to function correctly.

EXORB

Character: 'x'
Cell size: 2 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add series wire short on left side.
(2) add series wire ground on right side.
(3) add both short and ground on series wire.

Inputs: A (RCOL wire of left column)
A* (LCOL wire of left column)
B (RCOL wire of right column)
B* (LCOL wire of right column)
Outputs: Q (lower ROW wire)

The Q output is left true if (A and B*) is true, or (A* and B) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an exclusive-or function on the A and B columns. However, the cell can perform more general and-or type logic when two distinct signals are used for the B and B* inputs. Other cells connected to the Q output may influence the value of the Q output.

NOTE: The A and A* signals MUST be logical inverses in order for this cell to function correctly.

7.2.3. Multiplexor Cells

MMUXT

Character: 'h'
Cell size: 3 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: D0 (LCOL wire at bottom edge of cell)
D1 (RCOL wire at bottom edge of cell)
SELECT (ROW wire at bottom of cell)
Outputs: Q (RCOL wire at top edge of cell)

This cell is a two input multiplexor. The SELECT input determines which of the D0 and D1 inputs are connected to the Q output. When SELECT is TRUE, the D1 input is connected to the Q output. When SELECT is FALSE, the D0 input is connected to the Q output.

NOTE: This cell should be used only to transfer data from the inputs to the output. Simulation of the cell assumes that the cells are used in this manner.

MMUXB

Character: 'H'
Cell size: 3 rows, 1 column.
Placement: Any row, any column.
Modifiers: *(none)*

Inputs: D0 (LCOL wire at top edge of cell)
D1 (RCOL wire at top edge of cell)
SELECT (ROW wire at top of cell)
Outputs: Q (RCOL wire at bottom edge of cell)

This cell is a two input multiplexor. The SELECT input determines which of the D0 and D1 inputs are connected to the Q output. When SELECT is TRUE, the D1 input is connected to the Q output. When SELECT is FALSE, the D0 input is connected to the Q output.

NOTE: This cell should be used only to transfer data from the inputs to the output. Simulation of the cell assumes that the cells are used in this manner.

7.2.4. Transparent Latches

LATCH

Character: 'L'
Cell size: 6 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: D (RCOL wire)
G (ROW wire at bottom of cell)
G* (ROW wire at top of cell)
Outputs: Q (LCOL wire)

This cell is a transparent latch. The Q output will follow the D input when the G input is TRUE and the G* input is FALSE. Data will be latched when the G input is made false, and data may be retained for long periods by making G* FALSE after the data is latched by making G FALSE. The G* input should not be made FALSE at the same time that the G input is TRUE.

7.2.5. Static RAM

RAMCELL

Character: 'M'
Cell size: 5 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: R (RCOL wire of left column)
S (LCOL wire of right column)
ENABLE-TOP (ROW wire near top of cell)
ENABLE-BOT (ROW wire at bottom of cell)
Outputs: Q (interfaced to RCOL wire at bottom of RAM array)

Four RAM bits built into a single cell. ROW wires are used to enable the RAM cells for reading and writing. For a more complete description of the read, write, and precharge operations, see the description of the 'A' (RAMSENSE) cells.

RAMLCAP

Character: '['
Cell size: 5 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: (none)
Outputs: (none)

Cap for left side of a bank of RAM cells.

RAMRCAP

Character: ']'
Cell size: 5 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: (none)
Outputs: (none)

Cap for right side of a bank of RAM cells.

RAMLLCOR

Character: 'n'
Cell size: 1 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: (none)
Outputs: (none)

Cap for lower-left corner of a bank of RAM cells.

RAMINT

Character: 'm'
Cell size: 1 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: (none)
Outputs: (none)

Cap for lower edge of a bank of RAM cells. This cell interfaces the 4-column-wire memory grid to the 3-column-wire PPL grid. The two data outputs are interfaced to the LCOL and RCOL wires.

RAMLRCOR

Character: 'p'
Cell size: 1 rows, 1 column.
Placement: Any row, any column.
Modifiers: (none)

Inputs: (none)
Outputs: (none)

Cap for lower-right corner of a bank of RAM cells.

7.2.6. Sense Amplifiers and Caps

RAMSENSE

Character: 'A'
Cell size: 16 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) block-wide write-enable
(1) bit-wide write-enable

(mod 0 inputs)
Inputs: PRECHRG1 (first row above cell origin)
PRECHRG2 (sixth row above cell origin)
WRITE-ENABLE (thirteenth row above cell origin)
DATA0 (LCOL wire at top of cell)
DATA1 (RCOL wire at top of cell)
Outputs: 4 column wires of memory grid, at bottom of cell

(mod 1 inputs)
Inputs: PRECHRG1 (first row above cell origin)
PRECHRG2 (sixth row above cell origin)
WRITE-ENB0 (LCOL wire at top of cell)
WRITE-ENB1 (RCOL wire at top of cell)
DATA (SCOL wire at top of cell)
Outputs: 4 column wires of memory grid, at bottom of cell

Read/write sense amplifier for RAM cells. Must be placed above a block of RAM cells. These cells interface the normal PPL grid to the special grid used for the RAM cells, and perform all of the read, write and precharge operations required by the RAM cells. Each of these cells consists of three functional blocks: A write-enable circuit, a read precharge circuit, and a read sense amplifier.

The RAM grid uses four column wires, which are configured as two pairs of complimentary read/write signal wires. Reading and writing of data is accomplished by pulling one wire of a complimentary pair to a low voltage, while allowing the other wire to remain at a high voltage. During a WRITE cycle, the write-enable circuitry forces one wire low while forcing the complimentary wire high. Data is transferred from the column wires to the RAM cell when the ROW wire which enables the RAM cell is forced high. The low voltage column wire can then force the row-enabled RAM bit to change state, thus writing data into the cell. Before a READ cycle can be performed, both column wires of a complimentary pair must be precharged high. After precharging is completed and the precharge circuitry is disabled, a single row or the RAM cells can be enabled. A row-enabled RAM cell can then start to pull one of the column wires to a low level. This action is sensed by the read sense amplifier, which takes over the job of pulling the column wire low. Although it is theoretically possible for the RAM cell to pull a column wire to a low level, the sense amplifier is used because it greatly decreases the amount of time required to read data from the RAM cells.

The different cycles used in reading and writing the RAM cells are summarized below.

PRECHARGE: Precharge is required prior to each read cycle. Failure to precharge before read operations will result in loss of data in the RAM cells. PRECHARGE MUST NOT BE ASSERTED WHILE WRITE-ENABLE IS ASSERTED. Precharge is enabled for a short time, then disabled before enabling a row of RAM cells. After a precharge cycle is finished, a row of RAM cells can be enabled in order to read data from the RAM.

READ: Precharge must be performed to pull all column wires high. After precharge is disabled, one RAM row is enabled to read from memory. The enabled RAM cells will begin pulling only one of the complimentary data wires low. When the sense amplifier detects that a wire is being pulled low, the sense amplifier will rapidly finish pulling the wire low. After any read or write cycle, a new precharge cycle must be performed before the next read operation.

WRITE: Write-enable is asserted, which places write data on the RAM column wires. Data may then be written into RAM cells by enabling the appropriate row of RAM cells. The row-enable is asserted while the write-enable is asserted. WRITE ENABLE MUST NOT BE ASSERTED AT THE SAME TIME AS PRECHARGE. Precharge is required only before read cycles, thus multiple write cycles may be performed without precharge cycles in between.

SENSLCAP

Character: '{'
Cell size: 16 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) use with block-wide write-enabled sense amp.
(1) use with bit-wide write-enabled sense amp.

Inputs: (none)
Outputs: (none)

Cap for left side of a bank of sense amplifiers.

SENSRCAP

Character: '}'
Cell size: 16 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) use with block-wide write-enabled sense amp.
(1) use with bit-wide write-enabled sense amp.

Inputs: (none)
Outputs: (none)

Cap for right side of a bank of sense amplifiers.

7.3. Advanced PPL Cells (rarely used)

GNDCON

Character: 'g'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: [formed by summing values below]
(1) connect RCOL wire to ground.
(2) connect LCOL wire to ground.
(4) connect SCOL wire to ground.
(8) connect ROW wire to ground.
(16) connect SROW wire to ground.

Inputs: (none)
Outputs: (specified by modifier value)

Connect one or more wires directly to GND. Each wire is specified by the modifier value shown above. Combinations of wires are specified by a modifier that is the sum of the values specified for each wire that is to be grounded. A modifier value of ZERO will cause none of the wires to be grounded, which is equivalent to a BLANK cell.

VDDCON

Character: '^'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: [formed by summing values below]
(1) connect RCOL wire to ground.
(2) connect LCOL wire to ground.
(4) connect SCOL wire to ground.
(8) connect ROW wire to ground.
(16) connect SROW wire to ground.

Inputs: (none)
Outputs: (specified by modifier value)

Connect one or more wires directly to VDD. Each wire is specified by the modifier value shown above. Combinations of wires are specified by a modifier that is the sum of the values specified for each wire that is to be connected to VDD. A modifier value of ZERO will cause none of the wires to be connected to VDD, which is equivalent to a BLANK cell.

LCAP

Character: '('
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Caps off the left edge of a PPL module which is to be used in a custom circuit.

RCAP

Character: ')'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Caps off the right edge of a PPL module which is to be used in a custom circuit.

ROWVBUS

Character: '/'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Cell to route VDD bus along a row. Also used to cap off upper edge of PPL modules that are to be used in custom circuits. Included as part of pad cells automatically.

ROWGBUS

Character: '\'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Cell to route GND bus along a row. Also used to cap off lower edge of PPL modules that are to be used in custom circuits. Included as part of pad cells automatically.